RICHFIELD

FACULTY OF INFORMATION TECHNOLOGY

SOFTWARE DEVELOPMENT 500

1ST SEMESTER ASSIGNMENT

Name & Surname: Nhlamulo Shaun Mashaba **ICAS No:** 401414140

Qualification: DIT **Semester:** 1 **Module Name:** Software Development

Date Submitted: 2024/04/29

ASSESSMENT CRITERIA	MARK ALLOCATION	EXAMINER MARKS	MODERATOR MARKS
MARKS FOR CONTENT			
QUESTION ONE	40		
QUESTION TWO	30		
QUESTION THREE	20		
TOTAL	90		
MARKS FOR TECHNICAL ASPECTS			
TABLE OF CONTENTS	2		
LAYOUT AND SPELLING	3		
REFERENCES	5		
TOTAL	10		
TOTAL MARKS FOR ASSIGNMENT	100		
Examiner's Comments:			
Moderator's Comments:			
Signature of Examiner:		Signature of Moderator:	

Contents

Question One 3

Question Two 5

References: 8

Question One

The significance of using modules in programming is that it helps avoid similar poor-planning scenarios because it becomes easy to lose track of what specific code does or to introduce code that should be rewritten hence using modular programming ensures quality of a specific piece of code (TinyMCE Blog, n.d.). Modular programming is much easier to read because it allows developers to organize code into separate and reusable logical components based on their functionality which also makes it easier to manage the code (AIST Global, n.d.). Modules can be created independently then integrated into other programs hence coding this way encourages code reusability as modules can be adapted and reused across different projects and encourages collaboration among colleagues as the code is easily interpreted by others (Bit Blog, n.d.)

An example of modular programming being effective, using the example of a CMS, is that the management of articles, images and users can be implemented as separate modules.

For the article module/component it can handle all the CRUD operations which are basically, creating, reading, updating and deleting of any articles. This module can handle all the logic related to articles (Tateeda, n.d.).

For the images module/component it can be used for uploading, storing and retrieving images with the CMS. we can provide functions for image manipulation like resizing images, handling file formats and ensuring efficient storage (Stackify, n.d.)

For the user Module/component we can manage user authentication, registration, profile management and access control. we can also include functions for validating user credentials (i.e. passwords, username), managing user sessions and enforcing security policies (Toptal, n.d.)

There are several principles that can be used when designing high-quality programs, these act like rules for creating computer programs that work well and are easy to manage whilst ensuring that their programs are organized, efficient and easy to understand.

- SOLID Principles: Encourage modular, flexible, and maintainable software design
    - Single Responsibility Principle (SRP): this principle emphasizes that a module should have only one reason to change - a well-defined responsibility. This promotes code clarity and makes it easier to manage the code. (TinyMCE Blog, n.d.)
    - Open-Closed Principle (OCP): this principle encourages software entities to be open for extension but closed for modification. This means you should be able to add new functionality to a module without altering its code. For

example, we can add support for multimedia content without modifying the core functionalities (AIST Global, n.d.)
    - Liskov Substitution Principle: this principle states that when a derived class is used in place of its parent class, everything should work smoothly. There shouldn't be a reason to worry about unexpected errors or behaviours just because you're using a more specific version of the class (Bit Blog, n.d.)
- KISS (Keep it Simple Stupid):
    - This principle encourages the use of simple, well-known solutions whenever possible, instead of building complex algorithms from scratch. For example, when sorting a list, it's easier to use a built-in sort function rather than creating a new function. It prioritizes readability and maintainability over 'cleverness'. It focuses on the core of what required rather than the extras by breaking down complex problems into smaller, manageable task sand tackle them one at a time. (Tateeda, n.d.)

- DRY (Don't repeat yourself):
    - This principle encourages one to avoid tedious and duplicative efforts. it encourages one to extract duplicative code into a function or method then reuse it whenever necessary. It uses abstractions like modules/functions or methods to encapsulate and share common features and functionality (Stackify, n.d.)

There are specific practices one can deploy to contribute to the improvement of code maintainability and efficiency, in the context of a CMS, the following can contribute to the improvement of code:

- Modularity:
    - We can divide the different components of the CMS into modules that can be responsible for specific functionalities such as article management, image handling, user authentication and content publishing. We can implement each module as an independent unit with a well-designed interface and functionalities. (Toptal, n.d.)
- Consistency:
    - We can ensure that we maintain consistency within the code by using the same naming conventions (i.e. camelCase, new_var) across all modules. We can ensure that this is done by enforcing guidelines and adopting a coding standard. This has many benefits like code readability, maintainability and improved teamwork (TinyMCE Blog, n.d.).
- Documentation:
    - We can document the CMS codebase to facilitate understanding, maintenance and collaboration among developers. Things like comments to explain the purpose of certain sections of the code. We can create a separate document that strictly outlines the architecture, data models, API endpoints and configuration options of the CMS (AIST Global, n.d.).

Question Two

Enter subjects passed and the grades of subjects

Variables:
UserName
numberOfSubjects
subjects
gradeOfSubjects
scienceSubjects
isEligibleForBSC
isEligibleForDIT
isEligibleForHCIT
passLevel
grade

Start

Display welcome message: "Welcome to Richfield College Enrolment Program"

Print "Enter your name:"
Read userName

Print "Enter number of subjects passed:” // Example: 7
Read numberOfSubjects

Initialize subjects as an empty list: subjects = []

For i in range(numberOfSubjects):
  Print "Enter Subject:"
  Read subject
  Append subject to subjects list: subjects.append(subject)

Print "Grades of each subject:"

Initialize grades as an empty list: grades = []

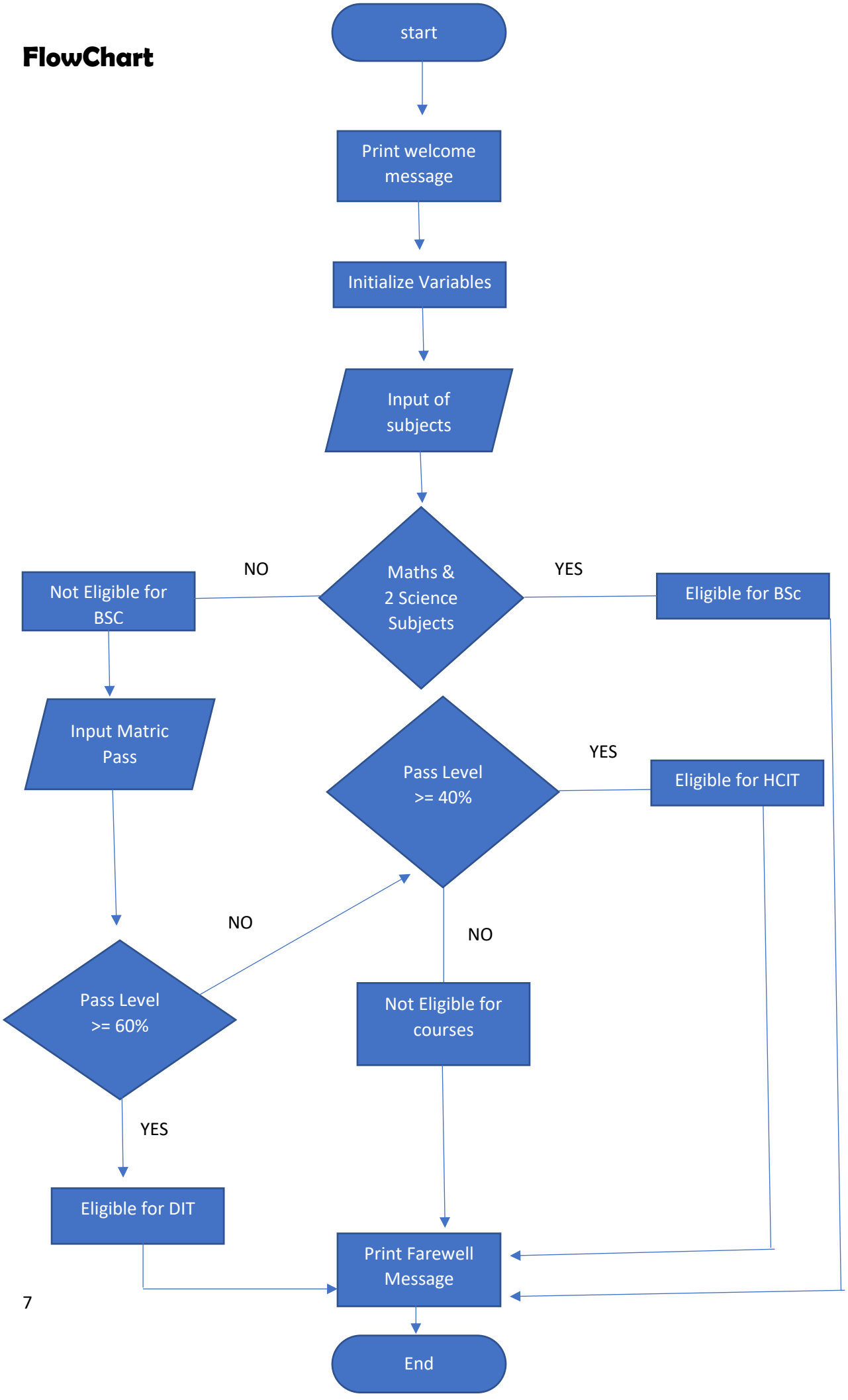
For i in range(numberOfSubjects):
  Print "Enter Grade for", subjects[i], ":" // Example: 66
  Read grade
  Append grade to grades list: grades.append(grade)

Initialize scienceSubjects counter: scienceSubjects = 0

For subject in subjects:

```
    If "science" in subject:
        Increment scienceSubjects counter: scienceSubjects += 1

Initialize eligibility flags:
isEligibleForBSC = False
isEligibleForDIT = False
isEligibleForHCIT = False

If "Maths" in subjects and scienceSubjects >= 2:
    Set isEligibleForBSC to true

Print "Enter your matric pass level:"  // Example: Senior Certificate, National Certificate
Read passLevel

If passLevel is "Senior Certificate":
    Set isEligibleForHCIT and isEligibleForDIT to true
Else if passLevel is "National Certificate":
    Set isEligibleForHCIT to true
Else:
    Set isEligibleForBSC, isEligibleForDIT, and isEligibleForHCIT to false

If isEligibleForBSC is true:
    Display "Congratulations, you are eligible for BSC in IT"
Else if isEligibleForHCIT is true:
    Display "Congratulations, you are eligible for HC in IT"
Else if isEligibleForDIT is true:
    Display "Congratulations, you are eligible for DIT"
Else:
    Display "You are not eligible for any course"

Print "Thank you,", userName, "for enrolling in Richfield College."
```

FlowChart


start
Print welcome message
Initialize Variables
Input of subjects
Maths & 2 Science Subjects
NO
YES
Not Eligible for BSC
Eligible for BSc
Input Matric Pass
Pass Level >= 40%
YES
Eligible for HCIT
NO
NO
Pass Level >= 60%
Not Eligible for courses
YES
Eligible for DIT
Print Farewell Message
End

References:

AIST Global. (n.d.). Importance of modular programming. Retrieved from https://aist.global/en/importance-of-modular-programming

Bit Blog. (n.d.). Software design principles. Retrieved from https://blog.bit.ai/software-design-principles/

Stackify. (n.d.). 7 steps to improve code quality. Retrieved from https://stackify.com/7-steps-to-improve-code-quality/

Tateeda. (n.d.). Fundamental principles of good software design. Retrieved from https://tateeda.com/blog/fundamental-principles-of-good-software-design

TinyMCE Blog. (n.d.). Modular programming principle. Retrieved from https://www.tiny.cloud/blog/modular-programming-principle/#:~:text=Modular%20programming%20usually%20makes%20your,smaller%20and%20easier%20to%20understand.

Toptal. (n.d.). CMS web design. Retrieved from https://www.toptal.com/designers/ui/cms-web-design